MEC 529 Final Project: Baxter Robot Motion Plan

Anisur Rahman

May 14, 2020

Abstract

A motion planning algorithm to reach a door knob, turn it, and then open the door using the right arm of a Baxter Robot was created in MATLAB. Using Screw Linear Interpolation (ScLERP) ensured that the position and orientation of the end effector throughout the motion conformed to the $SO(3)$ constraint intrinsic to motion about a revolute joint in $\mathbb{R}^3$ for the parts of the motion that required it. The error for each final configuration was less than 1mm in distance. Additionally, the final error for rotation was less than 0.001.

1 Introduction

The arm of a Baxter Robot is a redundant mechanism with 7 D.o.F. The purpose of this project was to create a motion planning algorithm for the right arm of a Baxter Robot to complete a task. In order to do so, ScLERP was used in conjunction with Velocity Inverse Kinematics. The task chosen was opening a door by first reaching the door knob, turning it, then swinging the door open. An error criteria of 0.001 was chosen for both distance and orientation. The initial position and orientation of the right edge of the door knob with respect to the world frame, $\{W\}$, was determined by computing the forward kinematics of arbitrary input angles to the robot arm. This ensured that the goal destinations were in fact valid for the arm. There are three different goal tasks to ensure that the code works for different types of motion.

2 Problem Statement

The right arm of a Baxter Robot needs to reach a door knob whose position and orientation with respect to the world frame can be described using the transformation matrix, $g = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix}$, where $g \in SE(3)$, $R \in SO(3)$, $p \in \mathbb{R}^3$, 0 is a 1×3 vector, and 1 is a scalar. R is the rotation matrix generated from rotation about an axis in $\mathbb{R}^3$. The vector p is the displacement between the origin of the world frame and the origin of the door knob's frame. The $1.2cm$ door knob's orientation was rotated by 1.18 radians about the axis $\begin{bmatrix} 0.8149 & 0.1807 & 0.5508 \end{bmatrix}^T$ and displaced by $\begin{bmatrix} 0.573m & 0.2886m & -0.0740m \end{bmatrix}^T$ with respect to the world frame. Once the door knob was reached, the end effector needed to rotate about its x-axis to turn it. Finally, the end effector rotated about the z-axis coinciding with door hinge to open it. The door knob measured $30cm$ from the hinge joint of the door. Because two of the motions have $SO(3)$ constraints, it is important that the orientation and position of the end effector be consistent with those constraints throughout the respective motions. The intial angle for all the joints of the arm will be 0^o. The arm is orientated 45^o CCW with respect to X^W, 45^o CW with respect to Y^W, and 90^o CCW with respect to Z^W.

3 Solution Approach

Since the Baxter Robot's arm is a reduntant mechanism., inverse kinematics cannot be used to solve for joint angles at discrete points between the two paths. Even for non-redundant mechanisms, this approach is generally not a good idea because inverse kinematics usually leads to multiple solutions for joint angles. The joint angles found between two points in the path may not belong to the same solution set, which will cause the alogorithm to fail because joint angles will have to instantaneously change by large amounts. To address this issue, Velocity Inverse Kinematics will be used to map solutions from the task space to the joint space. The steps for the algorithm are:

1. The present and final configurations, g_0 and g_f respectively, will be converted to dual quaternion representation: $D = A_r + A_d$
2. ScLERP will be used to compute the next configuration in the task space using the formula:

$$D(\tau) = D_0 \otimes (D_0^* \otimes D_f)^\tau, \text{ where } \tau = 0.01$$

3. The pose, $\gamma(t+1) = \begin{bmatrix} p_\tau \\ q_\tau \end{bmatrix}$, of $D(\tau)$ is extracted from the dual quaternion representation. $\gamma(t) = \begin{bmatrix} p \\ q \end{bmatrix}$ is found from D_0

4. Create the J_1 matrix:

$$J_1 = \begin{bmatrix} -q_1 & q_0 & q_3 & -q_2 \\ -q_2 & -q_3 & q_0 & q_1 \\ -q_3 & q_2 & -q_1 & q_0 \end{bmatrix}$$

   Next, create the J_2 matrix:

$$J_2 = \begin{bmatrix} I & 2\hat{p}J_1 \\ 0 & 2J_1 \end{bmatrix}$$

5. Calculate $B(\theta)$ using the pseudo-inverse of the J^s matrix:

$$B(\theta) = (J^s)^T(J^s(J^s)^T)^{-1}J_2$$

6. The next set of joint angles $\theta(t+1)$ is calculated using the equation:

$$\theta(t+1) = \theta(t) + \beta B(\theta))(\gamma(t+1) - \gamma(t))$$

   β is dynamically adjusted throughout the loop to ensure that the max step in θ for all the joints was $\leq 0.5^o$ for each iteration

7. The $FK(\theta(t+1))$ is computed to to map the joint angle solution to the task space and determine the g matrix for the next iteration.

8. The error conditions, $d = \sqrt{(x_f - x)^2 + (y_f - y)^2 + (z_f - z)^2} \leq 0.001$ and $min\{\|q - q_f\|, \|q + q_f\|\} \leq 0.001$ are checked for the output of the previous step. If both conditions are met, the goal is reached and the loop continues for the next goal position. If either error is too high, $\theta(t) \leftarrow \theta(t+1)$ and $g(t) \leftarrow g(t+1)$ and the loop goes back to Step 1.

9. Complete until all goal poses are met

4 Results

All three goal positions were met while satisfying the error criteria previously stated. The time needed for the iterative portion of the code was approximately 4.13 seconds. The path and orientations of the end effector are shown in Figure 1. After reaching the door knob, the next two motions were rotational about an axis in $\mathbb{R}^3$. During these motions, the position and orientation of the end effector are constrained. Because ScLERP is used, the simulated motions abided by these constraints and produced a realistic motion. An animation of the end effector path is provided in the submission folder.

The θ values used to create the path seen in Figure 1 are shown in Figure 2. Every time a new

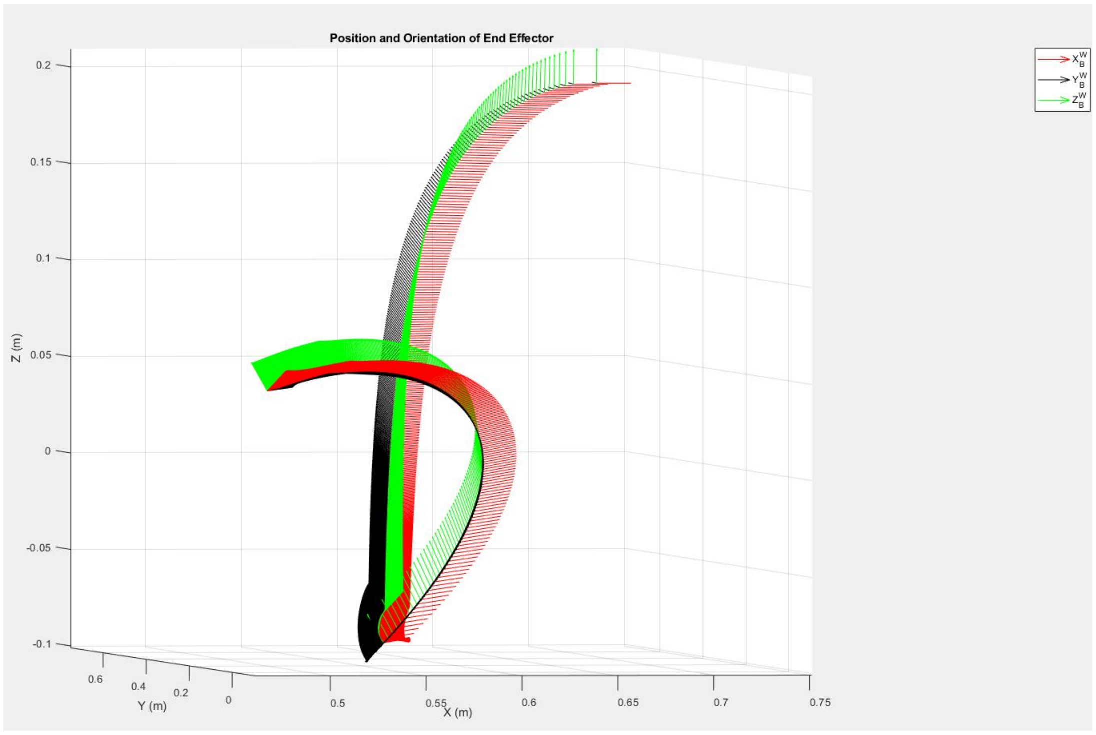

Figure 1: End effector path and orientation

goal is presented, the joint angles change rapidly at first. The change in angle becomes slower as the iterations grow because the error shrinks so less movement is necessary. The iteration numbers where the intermediate goals were met are 1139 and 2401. These are where the θ curves all begin to to grow quickly again because the end effector is working to a new goal. The total number of iteration it takes to complete the full task is 4196. The change in error over time is shown in Figure 3.

The trend of Figure 3 is related to the trend of Figure 2. The error is always greatest at the beginning of every path. Generally, the robot gets closer to its goal with each iteration, decreasing error. Once the desired position and orientation are reached, the error increases again because there is a new goal. The lowest error points correlate to the flattest parts of the θ plot. This is because the lower the error is, the less the joint angles have to change to correct it.

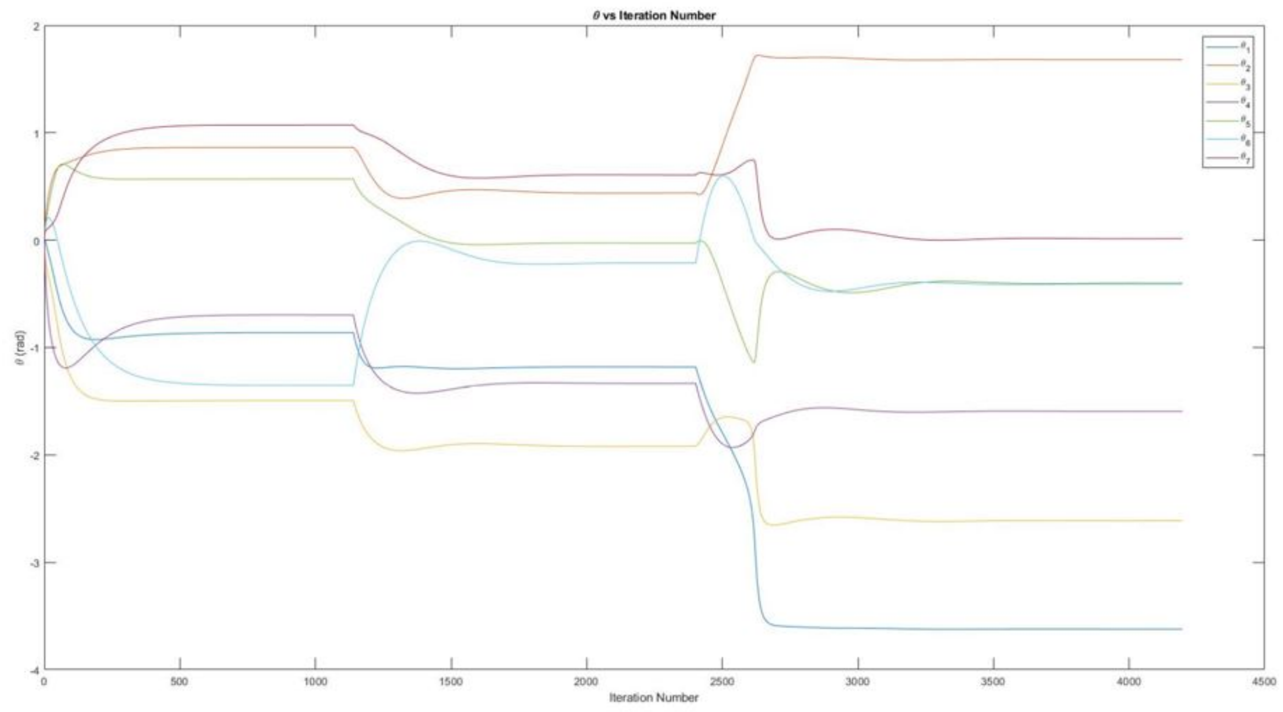

Figure 2: θ values for each joint over course of the end effector motion

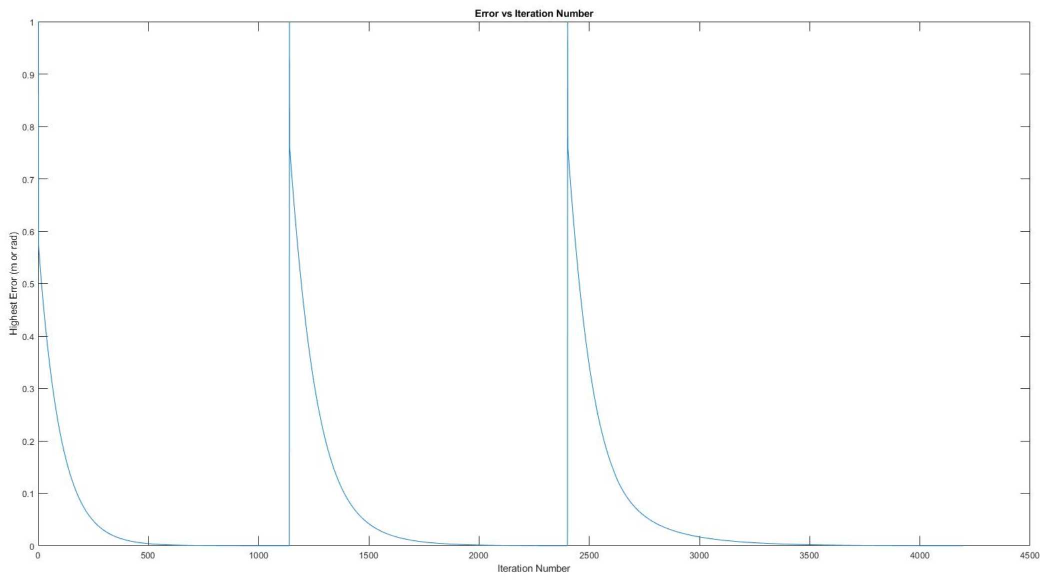

Figure 3: Error values over the course of the end effector motion

5 Conclusion

The analysis provided insight into the effectiveness of ScLERP metehod for motion planning. Two of the three motions had $SO(3)$ constraints and the ScLERP inherently satisfied these conditions without the need for additional programming. Because the Baxter Robot's arm is a redundant mechanism, it was important to have a method to determine $(J^s)^{-1}$. The pseudo-inverse method learned in lecture was used and proved effective for an application with the chosen error tolerances. The code did not take into account collision avoidance. For applications in real-life, this would be necessary because parts of the arm other than the end effector may have been in contact with the door or even passed through it. A joint space approach for collision avoidance could be used since β can be adjusted when a certain set of joint angles is undesirable. Additionally, joint angle limits were not considered for this code. This is a real world factor that would need to be considered in when implementing the motion plan.

6 Appendix

```
clear all
clc

%Baxter parameters

w1r = [0;0;1];
w2r = [-1/sqrt(2);1/sqrt(2);0];
w3r = [1/sqrt(2);1/sqrt(2);0];
w4r = w2r;
w5r = w3r;
w6r = w2r;
w7r = w3r;


l0=270.35/1000;
l1=69/1000;
l2=364.35/1000;
l3=69/1000;
l4=374.29/1000;
l5=10/1000;
l6=229.53/1000;

q1=[0,0,l0]';
q2=[l1/sqrt(2),l1/sqrt(2),l0]';
q3=q2;
q4=[(l1+l2)/sqrt(2),(l1+l2)/sqrt(2),l0-l3]';
q5=q4;
q6=[(l1+l2+l4)/sqrt(2),(l1+l2+l4)/sqrt(2),l0-l3-l5]';
q7=q6;

gst0=[eye(3) [(l1+l2+l4+l6)/sqrt(2) (l1+l2+l4+l6)/sqrt(2) l0-l3-l5
   ]';0 0 0 1];

axis_joints=[w1r w2r w3r w4r w5r w6r w7r];
q_joints=[q1 q2 q3 q4 q5 q6 q7];
for i=1:7
    type_joints(i,1)="revolute";
end

%The handle coordinates are generated using random angles to ensure
   that
%the final destination is actually reachable by the robot arm
```

```
gf(:,:,1)=BaxterFK([-pi()/4 pi()/13 pi()/9 pi()/3 -pi()/2.5 -3*pi()
   /5 2*pi()/7]);

%Once the handle is reached, a 12cm doorhandle is turned 180 deg
   clockwise
%The orientation will change by 90 deg clockwise and the final
   position
%will be translated 12cm to the left and down relative to the door
   knob's
%frame

gf(:,:,2)=gf(:,:,1)*[AxisAngle_to_Rot([1 0 0],-pi()/2) [0 -0.012
   -0.012]';0 0 0 1];

%After the handle is pulled, the door needs to be opened which is
%essentially rotating about the z axis by 90 degrees relative to the
%handle/door orientation. The door is 30 cm is long so x and y
   coordinates
%will change negatively by that amount since the rotatoin is CCW (
   pulling)

gf(:,:,3)=gf(:,:,2)*[AxisAngle_to_Rot([0 0 1],-pi()/2) [-0.30 -0.30
   0]';0 0 0 1];

%Dual Quaternion Rep for each configuration
for i=1:3
    Df(:,:,i)=GtoDQ(gf(:,:,i));
end
%All angles initially 0
theta(1:7,1)=0;
g0=BaxterFK(theta); %generate g matrix from joint angles


D0=GtoDQ(g0); %Dual Quaternion rep of g0

for i=1:3
    p_final(1:3,i)=gf(1:3,4,i); %goal positions of the EE
    q_final(1:4,i)=Df(1:4,1,i);
end



C=1;
k=1;
beta=1;
```

```
tic
for i=1:3
    E(k)=1; %k will be looped throughout three goals so E(k) will
       have to be reset
    %above the logic condition when the next goal is intially being
       worked towards
    while E>0.0001 %Error tolerance
        Di=QuatExponent(DQMultiply(DQInverse(D0),Df(:,:,i)),0.01);
        DI=DQMultiply(D0,Di); %interpolation step


        g_now=DQtoG(D0); %generate gamma for interpolation step
        p_now(1:3,1)=g_now(1:3,4);
        q_now(1:4,1)=D0(:,1);
        gamma_now=[p_now;q_now];

        x(k)=p_now(1); %will be used to plot EE path later
        y(k)=p_now(2);
        z(k)=p_now(3);

        g_interp=DQtoG(DI); %G matrix from interpolation step
        q_next(1:4,1)=DI(:,1); %generate gamma for interpolation
           point
        p_next(1:3,1)=g_interp(1:3,4);
        gamma_next=[p_next;q_next];

        q0=q_now(1); %Used for J1 matrix
        q1=q_now(2);
        q2=q_now(3);
        q3=q_now(4);

        J1=[-q1 q0 q3 -q2;-q2 -q3 q0 q1;-q3 q2 -q1 q0];
        J2=[eye(3) 2*skew_symmetric(p_now)*J1;zeros(3) 2*J1];
        Js=SpatialmanipJac(axis_joints,q_joints,type_joints,theta(:,
           k));
        B=Js'*inv(Js*Js')*J2;

        diff=gamma_next-gamma_now; %direction of change for theta

        if abs(max(diff))>0.01745/2 %linearization won't hold for
           steps too large so 1/2 degree is used
            beta=0.0175/2/abs(max(diff)); %this insures that the max
               step will always be less than 1/2 degree
        else beta=1; %if the step is smaller that is fine
        end
        theta(1:7,k+1)=theta(:,k)+beta*B*diff; %next theta values

        g_test=BaxterFK(theta(1:7,k+1)); %find the actual position
           from FK
        p_test(1:3,1)=g_test(1:3,4);
        q_test(1:4,1)=Rot_to_Quat(g_test(1:3,1:3));
        Rorientation(1:3,1:3,k)=g_test(1:3,1:3);

        distance=norm(p_test-p_final(:,i)); %error in position
        qq=min([norm(q_test-q_final(:,i)) norm(q_test+q_final(:,i))
           ]); %error in rotation
        E(k+1)=max([distance qq]); %max of the two errors is used

        D0=GtoDQ(g_test); %D0 loops to next value
        k=k+1; %advances indices
    end
    step(i)=k-1;
end
toc

%The large block of commented code is used to animate the path. This
    takes
%very long due to the large number of points. A video file of the
   path is
%provided in the submission .zip file.

% view(-30,-23)
% xfin=p_final(1,:);
% yfin=p_final(2,:);
% zfin=p_final(3,:);
% for i=1:3
%     scatter3(xfin(i),yfin(i),zfin(i),'g');
%     hold on
% end
%Animate tool path
% curve=animatedline('LineWidth',2);
% set(gca,'XLim',[min(x) max(x)],'YLim',[min(y) max(y)],'Zlim',[min(
   z) max(z)])
% hold on
% for i=1:length(x)
%     addpoints(curve,x(i),y(i),z(i));
%     head=scatter3(x(i),y(i),z(i),'filled','MarkerFaceColor','r','
   MarkerEdgeColor','r');
%     drawnow
%      xlabel('x (m)')
%      ylabel('y (m)')
%      zlabel('z (m)')
%      title('End Effector Path')
%      delete(head)
% end

%Plot theta values
for i=1:k
    K(i)=i;
end

figure()
plot(K,theta)
title('\theta vs Iteration Number')
xlabel('Iteration Number')
ylabel('\theta (rad)')
legend('\theta_{1}','\theta_{2}','\theta_{3}','\theta_{4}','\theta_
   {5}','\theta_{6}','\theta_{7}')

%Plot Errors
figure()
plot(K,E)
xlabel('Iteration Number')
ylabel('Highest Error (m or rad)')
title('Error vs Iteration Number')


for i=1:3
Check(:,:,i)=BaxterFK(theta(:,step(i))); %validate theta solution
   set
end
%theta solutions validated


Rorientation=Rorientation/50; %decrease length of vectors for visual
    pruposes. It is known that their length should be one
Pos=[x;y;z];

figure()
for i=1:length(x)
quiver3(x(i),y(i),z(i),Rorientation(1,1,i),Rorientation(2,1,i),
   Rorientation(3,1,i),'r') %Xwb
hold on
```

```
quiver3(x(i),y(i),z(i),Rorientation(1,2,i),Rorientation(2,2,i),
   Rorientation(3,2,i),'k') %Ywb
hold on
quiver3(x(i),y(i),z(i),Rorientation(1,3,i),Rorientation(2,3,i),
   Rorientation(3,3,i),'g') %Zwb
hold on
end
view(-30,-23)
xlabel('X (m)')
ylabel('Y (m)')
zlabel('Z (m)')
title('Position and Orientation of End Effector')
legend('X^{W}_{B}','Y^{W}_{B}','Z^{W}_{B}')
axis equal

fprintf('\nThe g matrices for the theta values at each goal are:\n')
display(Check)
fprintf('\nThe goal g matrices are:\n')
display(gf)
fprintf('\nSolutions for joint angles at goal position are validated
   \n')
```